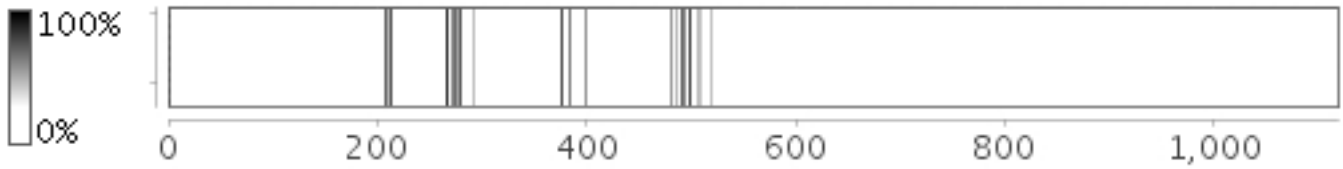
PaperPass旗舰版检测报告
简明打印版

比对结果（相似度）：
总　体：12 %　*（总体相似度是指本地库、互联网的综合比对结果）*
本地库：8 %　*（本地库相似度是指论文与学术期刊、学位论文、会议论文、图书数据库的比对结果）*
期刊库：5 %　*（期刊库相似度是指论文与学术期刊库的比对结果）*
学位库：6 %　*（学位库相似度是指论文与学位论文库的比对结果）*
会议库： 1 %　*（会议库相似度是指论文与会议论文库的比对结果）*
图书库：2 %　*（图书库相似度是指论文与图书库的比对结果）*
互联网：6 %　*（互联网相似度是指论文与互联网资源的比对结果）*

编　号：592FCB3D18D3CXL0I
版　本：旗舰版
标　题：基于MIPS平台的智能小车
作　者：张亮
长　度：29543 字符(不计空格)
句子数：1118句
时　间：2017-6-1 16:07:25
比对库：学术期刊、学位论文（硕博库）、会议论文、图书、互联网资源
查真伪：http://www.paperpass.com/check

句子相似度分布图：



本地库相似资源列表（学术期刊、学位论文、会议论文、图书）：
1. **相似度：2 % 篇名:《基于OpenCV的变电站监控视频智能分析》**
   来源：学术期刊 《新型工业化》 2016年8期 作者: 王华键 王一智 董芸含 刘一宸 俞晓婷 顾玖
2. **相似度：1 % 篇名:《视频图像中的多姿态人脸检测》**
   来源：学位论文 湖南大学 2014 作者: 熊英
3. **相似度：1 % 篇名:《基于超声波与图像识别的盲人导航眼镜研究与实现》**
   来源：学位论文 电子科技大学 2016 作者: 胡娟
4. **相似度：1 % 篇名:《基于Android平台的视觉手势识别研究》**
   来源：学位论文 西安电子科技大学 2012 作者: 王赞超
5. **相似度：1 % 篇名:《基于树莓派的农作物低空观测系统设计》**

来源：学术期刊 《吉林大学学报（信息科学版）》 2015年6期 作者: 张怀柱 姚林林 沈扬 姚欣宜

6. 相似度：1% 篇名:《人脸检测设计与实现》
来源：学术期刊 《福建电脑》 2012年10期 作者: 胡云琴 梁春美

7. 相似度：1% 篇名:《基于人脸识别的中学课堂考勤系统的设计》
来源：学术期刊 《科技传播》 2015年15期 作者: 陈鸿飞 严忱君 俞宝福

8. 相似度：1% 篇名:《基于肤色和Haar方差特征的人脸检测》
来源：学术期刊 《计算机工程与科学》 2015年1期 作者: 李燕 王玲

互联网相似资源列表：

1. 相似度：6% 标题：《树莓派+传感器：创建智能交互项目的实用方法、工具及最佳实践全文阅读_...》
http://yuedu.baidu.com/ebook/a366d95f02d276a201292e36?pn=6&pa=16&isFromWenku=1

2. 相似度：1% 标题：《级联分类器 - twinkle_star1314的博客...》
http://blog.csdn.net/twinkle_star1314/article/details/53291624

全文简明报告：

本科毕业设计（论文）

题目

基于MIPS平台的智能小车

学生姓名

张亮

学 号

201317040218

教学院系

理学院

专业年级

电子信息科学与技术2013级

指导教师

周云旭

职 称

实验师

单 位

西南石油大学

辅导教师

职 称

单 位

完成日期

2017

Southwest Petroleum University

Graduation Thesis

Smart car based on MIPS platform

Grade: 2013

Name: Zhang Liang

Speciality: Electronic Information Science and Technology

Instructor: Zhou YunXu

School of Sciences

2017-6-1

摘要

{45%：树莓派是为学习计算机编程教育而设计的一块只有信用卡大小的基于Liunx系统的微型电脑[1]。} 本文主要基于树莓派的应用程序开发，研究相关感器的工作原理和摄像头视频传输的特性以及网络编程方法，{47%：利用面向对象编程语言 Python编写了服务器应用程序。} 用PyQt设计了一个客户端应用程序。 最终实现计算机客户端通过无线网络远程对树莓派小车进行控制， 远程接收经过树莓派处理的传感器数据和摄像头的图像并将这些数据和图像上传至物联网云平台 Yeelink， 实时数据处理的同时，提供安全可靠的状态监控，确保数据只在允许的范围内共享。

本文主要研究内容： {44%：首先研究基于linux操作系统的树莓派，通过命令行终端对操作系统进行控制，} 深入分析研究该微型电脑的通用输入输出引脚功能，利用GPIO库和Python编程语言访问操作这些引脚， {58%：实现接收外围设备数据以及控制外围设备的目的。} 然后，研究常用传感器的工作原理，并通过Python编写的程序让多个传感器集成在树莓派下协调工作，完成数据采集与通信。 最后，研究树莓派互联网接入协议，完成物联网建设，使其成为数据采集终端。 基于Python编程分别实现USB接口和CSI接口的摄像头视频图像采集和传输，并结合OpenCV和树莓派，学习基于计算机视觉的图像处理、识别方法。

关键词： 树莓派； 传感器； Linux； Python； OpenCV

Abstract

Raspberry pi is designed for learning computer programming education a credit card sized based on micro computer [1]. Liunx system in this paper the main application development of raspberry pie based on the working principle and characteristics of camera video transmission of sensor and network programming method, using object- oriented programming language Python server application. A client application program with PyQt. Finally we realize the computer client through a wireless network of remote control car number of raspberry pie, the image sensor receives the remote processing according to the raspberry pie and the camera and the image data and uploaded to the Internet of things cloud platform Yeelink, real- time data processing and provide security monitoring reliable, secure data sharing only in the permitted range.

The main research contents of this paper: firstly, the Linux operating system based on raspberry pie, the operating system is controlled by the command line terminal, in- depth analysis of the micro computer universal input and output pins, the pins access operation using the GPIO library and the Python programming language, realize the receiving peripheral equipment data and control peripheral devices. Then, the working principle of the commonly used sensor, and through the Python program for multiple sensor integration in raspberry pie coordination work, the completion of data acquisition and communication. Finally, study on the raspberry pie Internet access protocol, to complete the construction of things, the data acquisition terminal. The video camera image acquisition and transmission Python programming to achieve USB interface and CSI interface based on OpenCV and combined with raspberry pie, learning image processing based on computer vision recognition method.

Keywords: Raspberry pi; sensor; Linux; Python; OpenCV

目录

1 绪论

1.1 选题目的及意义

随着过去几十年智能化技术和互联网技术的飞速发展，越来越快捷紧密的互联网技术与越来越高的智能化技术已融为一体， 这种趋势进而带动了汽车电子技术尤其是智能小车技术的发展，这些技术包含了多个学科， 如传感器技术、机械电子、生物工程等。 {43%：智能车从某种意义上是人工智能的一个发展方向，它被应用于多个领域。} 如农业种植、变电站巡视、外星探索、军队建设等。 所以发展智能车技术，有利我国的国防建设、科研探索等领域。

在国内，国家电网使用的变电站小车，能够自行对变电站内的设备进行监控和维护，保证了设备的安全运行，降低了工作人员的安全风险。 在国外，农作物小车不仅能够在无人操作下于田间运行，而且能利用高光谱镜头、激光探头等收集传感器数据，实时为农业种植者提供有价值的农作物信息。

智能化小车满足了人们的切实需求，这种趋势在未来将会越来越明显。 本智能小车设计正是顺应着这种发展的趋势而产生的，通过一个小型的电动车，搭载相应的电机驱动控制和舵机驱动控制板， 辅以超声波、温湿度等传感器以及两个摄像头，并结合一块树莓派组成，结构小巧，但能完成诸多功能， {41%：如环境探测、人脸识别、视频采集和远程手动控制行驶等。} 结合这些功能，小车能完成一些人无法完成的操作，比如在狭小的空间内进行搜索、探测，在有毒害的环境下进行勘测等。

1.2 国内外研究现状

1.2.1变电站巡视车

新西兰 Transpower公司负责从变电站通过高压输电线为新西兰各地的城镇和城市地区供电，2013年9月该该公司开始研发一个变电站巡视车， 其目的是为了实时评估变电站内的运行情况，并上报给变电站的工程师。 该公司的变电站小车，能够在其搭载的计算机上虚拟化变电站内的设备，并通过一个电脑桌面进行控制，这缩短了工作人

员对故障设备的响应时间。 小车能够在操作人员或工程师的控制下灵活运行，

它能够通过光谱摄像头等检测仪器，透视有着坚固外壳的变电站设备，进而查看内部情况，对设备内部进行定性检测。 同时，小车运行时不受恶劣气候和高压电危险的影响，操作人员可以命令小车实时关注变电站的特定设备， 当该设备出现任何意外故障时，小车自动发送回放视频和静态图片。

{56％：在我国，对于变电站小车的研究起步较晚。} {44％：2007年开始，变电站巡检小车体系结构被鲁守银博士提出[3]，小车的避障碍算法与动力学建模也基于该体系结构。} 小车的云台控制系统被曹雷等设计[5]，为小车上搭载云台产品提供了可靠性。 而着眼于低成本和低功耗的嵌入式变电站小车由矫德余等研制[4]，能够实现小车在变电站自主巡视。 {66％：王健远利用基于路径搜寻算法和智能寻迹方案[6]，对各检测点关联信息进行关联路径搜索，为智能巡检提供了新思路。}

总之，智能变电站小车经过近几年的发展，国内外的研究都取得了显著的突破，但智能化在小车上依然存有复杂多变性， {45％：导致变电站小车在实际应用中仍然存在局限。}

1.2.2 农业小车

{58％：智能农业小车在农业领域发挥着越来越突出的作用。} 它们通常被用来完成农作物的种植，如对农作物的种植，灌溉，施肥，监测和收获。

美国提出的农业小车系统侧重于实施所有这些种植过程，如通过使用Fire bird v小车种植洋葱作物[22]。 {41％：Fire bird v小车使用ATMEGA 2560作为主控制器，ATMEGA 8作为从控制器，IR，夹具布置等配件。} 小车通过使用传感器和夹持器布置种子种植在相应的位置来检测种植面积，剩余的农作物种植过程也能够按此方式自动完成。 该系统使用 AVR ATMEL STUDIO6.0和 AVR引导加载程序进行编程小车， ATMEL STUDIO6.0 s/ w用于写入编码， AVR引导加载程序作为 PC和 PC之间的接口，夹具布置用于种植、收获植物、喷洒肥料和农药。 这个小车可以帮助农民做好准确的耕作。

在印度，农业的主要问题是投入成本上升，熟练劳动力供应不足，缺乏水资源和不能对农作物进行实时监控。为了克服这些问题，智能化技术被用于农业，印度的Satish Kumar KN，Sudeep CS等人提出了一个多用途农业小车来实现精准灌溉。 小车除了连续监测作物和土壤条件外，还要灌溉，添加肥料和喷洒除草剂。 基于作物产量、土壤密集检测和环境等各种参数，小车能够合理分配水资源，帮助农民最佳利用资源和减少投入成本。

2015年，国际电气与电子工程协会的Patrick M. Piper和Jacob S. Vogel等人设计了一个智能土壤监测小车，小车配备了用于感测土壤水分和温度的史蒂文斯水螅探针 II和 GPS卫星导航， 能够迅速地对土壤数据的采集和自主地浏览一个领域并避免障碍。 它将收集一组给定点的土壤水分和温度数据，并将信息传回农场种植员。

Jayashree GC和Mahendran R等人通过图像分析提出了对水果进行分类和分级的方法[8]，研发了水果采摘小车。小车主要由机械摇臂、机械手、视觉处理系统组成，能够使用物理图像传感器获取水果图像， {43％：并使用专用计算硬件和软件进行图像采集、图像预处理、图像解释，对图像内的水果进行量化和分类，} 评估水果的质量和种类，完成对水果的采摘和分类。

1.3 本次毕业设计完成的主要内容

（1）研究树莓派通用的架构、标准的接口和外围设备，将传感器集成到树莓派电脑板。

（2）利用Linux和Python深入挖掘树莓派的功能，了解各种硬件和软件如何协调工作，并在树莓派上开发传感器和物联网项目。

（3）研究利用树莓派和传感器测量距离、监控温度和湿度、在线上传传感器数据以及视频图像。

（4）结合树莓派和OpenCV，研究计算机视觉的图像处理技术，并将一个摄像头和OpenCV库制作图像传感器进行人脸识别。

{45％：（5）深入探究树莓派的GPIO引脚以及如何通过Python和shell脚本访问这些引脚。} （6）利用基于Pyhon的网络编程，搭建树莓派小车服务器的运行环境。

{42％：（7）研究PyQt4的信号与槽机制，完成对操作界面的GUI图形编程。}

1.4 设计思路

图1.1小车模块框架图

为完成小车服务器的搭建，实现设计的诸多功能，首先需要把智能小车分解成这几个部分。 如图1.1所示，即树莓派主控模块、电源模块、电机驱动模块、传感器模块等，然后研究各部分模块的工作原理，最后编程实现相应功能。

2硬件与Haar分类器原理分析

2.1 树莓派GPIO

树莓派通过40个通用输入输出口，能够扩展各种外部设备，并且该通用输入输出口具备可编程功能。 {49％：如图2.1所示，树莓派40Pin引脚对照表。}

{58％：图2.1树莓派40Pin引脚对照表}

利用python-gpio库，在执行Python脚本时轻松地访问和控制树莓派的GPIO引脚，即可以通过编程控制GPIO管脚输出高低电平等。 此外，还有一个为GPIO上的Python SPI接口使用支持的库： spidev。

2.1伺服马达

GPIO输出信号只有开或关的数字信号。 可以通过编程来提供3V或0V的电压。 因此，对于软件，无法要求GPIO提供1V或2V的电压。 尽管有此限制，模拟电路可通过使用PWM（脉冲宽度调制）进行数字输出来驱动。

PWM技术基于信号平均原理。 如果信号占空比为10%，那么当信号被平均掉，其结果是两个数字极值10%的模拟量。 例如GPIO输出电压的最大值为5V，那么一个占空比为75%的重复数字信号，产生的平均电压计算如下[9]：

（2.1）

如果GPIO的输出信号占空比为10%，平均后的结果是Vavg = 0.5V。 平均效果有几种方法可以实现，例如对于一个灯泡，原件被接通脉冲加热，但不立即冷却，所以其亮度放映了平均电流； 对于直流电机，当电流消失，因为转动惯量保持转子旋转，直流电动机不会立即停止； 对于仪表，电流移除时，仪表指针不会立即回0.所以这些论效应都具有可被利用的平均效果。

2.2传感器

1.超声波传感器。 传感器向障碍物发出超生波，等待一段时间后就会接收到回波，其工作在超生波频率，该频率高于人类听觉的范围。 {51%：人类听觉的平均理论频率范围为20Hz~20KHz，所以人类的耳朵听不见它，并且能在短距离范围内进行精确测距。} {83%：超生波传感器会巧妙地生成声波脉冲，然后进行回波计算。} {78%：回波由同一个超声波传感器接收，通过计算发出信号和接收信号的时间间隔来确定距离[10]。}

{82%：如图2.2所示，HC_SR04超声波传感器在底板上有两个柱条，柱条上方覆盖有金属网，柱条一般是钢制或者其他等效材料制成。}

图2.2 HC_SR04超声波传感器

{53%：HC_SR04具有一个接收器和一个发射器。} {70%：更确切地说，有多个接收器和发射器，而该传感器在400 cm范围的测距精度接近3 cm.柱条之下是该传感器的控制电路，} {100%：用于完成所有任务，包括与树莓派进行通信。} 从该传感器引出了4个引脚： GND--接地引脚、ECHO--回波引脚、TRIG--触发引脚和VCC--供电引脚。 {88%：接地引脚与和5V供电引脚与树莓派相连。}

2.温湿度传感器。 {43%：如图2.3，DHT11组合封装了温度和湿度传感器，包含了一个电阻式热传感器（热敏电阻）和一个电容式湿度传感器。} 当环境温度稍有变化时，热传感器的电阻值会产生几百欧姆的改变，在0~55之间测量的温度值能达到2。 {42%：湿度传感器中排列着一些聚合物或金属氧化物，形成电容片。} 由于湿度的影响，电容的绝缘系数会发生改变，据此可以探测到湿度值的变化，在20%~80%湿度范围精度5%[11]。

图 2.3 DHT11 温湿度传感器

在树莓派上写入的用来读取DHT11传感器的用户空间软件使用直接寄存器来访问GPIO引脚。 DHT11信号使用固定的第一半50us，在此之后，该位取决于信号在高电平的持续时间。 因此，一个传输请求可以通过计算它能读取的低电平的信号次数来得到一个相对概念。 {42%：在此基础上，它会得到一个关于信号的短的和长的高电平信号的分割线位置。} DHT11数据格式如表2.4所示。

表2.4 数据格式

引脚说明：

VCC

供电 3.5V-5.5V DC

GND

接地，电源负极

N/A

空脚

数据格式：

相对湿度(16Bit)|摄氏温度（16Bit）|校验位（8Bit）

{48％： 树莓派通过GPIO引脚从输出变为输入的方式来控制传感器总线。} {43％： 总线闲暇时，上拉电阻作为一个输入电路连接，使总线变为高电平。} 当被要求时，传感器捕获总线并使其进入高电平或低电平。 最后，当控制器发出指令时，将该引脚设置为一个输出端，使GPIO引脚驱动总线。

3.气压传感器。 {50％： BMP180是用于消费应用的高精度 MEMS压力传感器，由压阻传感器，} {45％： 模数转换器和带有 E2 PROM和串行 I2 C接口的控制单元组成。} BMP180的工作压力和温度范围为300hPa 1100hPa，0850。 传感器的工作电压为3.3V。 该设备基于压电技术，具有EMC稳健性，高精度和线性度以及长期稳定性，并且被设计为通过于I2C总线，直接连接到移动设备的微控制器[12]。

控制设备发送特定数据以开始测量，然后经过一系列电平转换，数据位产生，最终得到结果值，并通过I2C接口读取结果值。 为了计算以为单位的温度和以hPa为单位的压力，必须使用校准数据。 而BMP180不提供校准数据，这些常数可以在软件初始化时通过I2C接口从BMP180的E2PROM中读出。

UP＝压力数据（1619位） UT＝温度数据（16位）

绝对海拔计算公式:

（2.2）p: 当前压强hpa p0: 1013.25hPa 标准大气压

2.3 Haar分类器方法分析

{95％： 人脸检测/Haar分类器，这个物体检测方法巧妙地使用了boosting算法。} 其形成如图2.5所示：

AdaBoost算法级联

积分图像算法

图2.5 Haar分类器

Haar分类器算法的要点如下：

使用Haar-like特征做检测。

{47%：如图2.6，子窗口如图(a)~a(e)将任意一个矩形放到人脸区域上，} {83%：然后，将白色区域的像素和减去黑色区域的像素和，得到的值即为人脸特征值，} {81%：若将矩形放到一个非人脸区域，则计算出的特征值应该和人脸特征值不相等，} {100%：所以这些方块的目的就是把人脸特征量化，以区分人脸和非人脸。}

图2.6 (a)~(e) 原始矩形特征

使用积分图（Integral Image）对Haar-like特征求值进行加速[14]。

{77%：积分图就是只遍历一次图像就可以求出图像中所有区域像素和的快速算法，大大的提高了图像特征值计算的效率，它能够描述全局信息的矩阵表示方法。} {92%：积分图的构造方式是位置（i，j）处的值ii(i，j)是原图像(i，j)左上角方向所有像素的和[15]：}

（2.3）

{76%：而Haar-like特征值即为两个矩阵像素和的差，同样使用积分图可以在常数时间内完成。}

{100%：使用AdaBoost算法训练区分人脸和非人脸的强分类器。}

利用 AdaBoost算法可以选择更好的矩阵特征组合，其实，矩阵特征组合即为上述提到的分类器， {40%：分类器将矩阵组合以二叉决策树的形式存储起来[14]。}

{100%：使用筛选式级联把强分类器级联到一起，提高准确率}

{91%：通过 Adaboost算法训练出了强分类器，然而在现实的人脸检测中，只靠一个强分类器还是难以保证检测的正确率，} {85%：需要训练出多个强分类器将它们级联，最终形成正确率很高的级联分类器：} Haar分类器。 {90%：级联强分类器的策略是，将若干个强分类器由简单到复杂排列，期望经过训练使每个强分类器都有较高检测率，而误识率可以放低[16]。} 如图2.7最终识别过程：

图2.7 Harr分类器识别过程

3 解决的关键问题

3.1 伺服电机转向角度计算

树莓派上的通用输出输入口能够编程实现脉冲的产生，而伺服电机可接收脉冲，并根据不同占空比的脉冲，带动转动电机转动不同的角度，达到控制伺服电机的目的。

{43%：关键需要确定输出脉冲的占空比与伺服电机转动角度之间对应的关系。} 如图3.1所示。

图3.1 pwm脉冲

本设计采用SG90型号的伺服电机，SG90向外引出三个引脚，分别为控制信号、电源和接地引脚。 {44%：电源模块接伺服电机的电源引脚，为其提供5v的直流电。} {67%：控制信号引脚需要输入一个频率为50Hz的方波

。} 根据方波不同的占空比，即高电平持续不同的时间，伺服电机的转动角度发生相应变化。 如表3.2所示：

表3.2 角度对照表

高电平持续时间(ms)	角度变化()	占空比(%)
0.5		
2.5		
1.0		
45		
5.0		
1.5		
90		
7.5		
2.0		
135		
10.0		
2.5		
180		
12.5		

设要转动的角度为r，每一度的高电平时间为 T，角度与电平持续时间的关系为： 0.5 [= 0.5 + r*T [= 2.5。 由表3.2得角度变化为180时，高电平持续时间2ms,

所以T =20ms/180= 2000us/180 = 11.1us/度，而任意一个角度r（0-180）的PWM脉冲宽度为： (500 + r*T)us。

综上，得出占空比与角度的计算公式:

（3.1）

由输入的的角度r，计算出占空比值，最后将占空比传入到Python GPIO模块中的ChangeDutyCycle()方法中，实现对伺服电机的转动角度的控制。

3.2 直流电机转向变换

使用直流电机驱动树莓派小车的行驶，但驱动直流电动机面临的问题之一是它们有时需要具备方向运转的能力。 为此电流必须反转。 这就需要额外的硬件支持。 {61％：H桥驱动器可以用来驱动可逆直流电动机或双极步进电机。}

图3.3 L298全桥驱动器

如图3.3示，L298集成电路的框架图。 H桥由8个驱动晶体管组成。 {61％：两个驱动电机分别接L298驱动器的OUT1、OUT2与OUT3、OUT4引脚。}

当Q1与Q4导通时，电机被启动。 当其他晶体管导通时，Q2和Q3保持关闭。 如果允许Q1和Q2在同一时间导通，6V动力电源接地将短路。 与逻辑门驱动这些晶体管以防止短路。

{58％：Q1和Q4同时打开，驱动电机的电流从左至右流过。} 关闭所有晶体管，没有电流。 {47％：打开Q3和Q2，电流从6V动力电源接地，这段时间在电机中电流从右至左流。} 控制晶体管对，电流可以在一个方向或其他方向流动。

表3.4 引脚功能

OUT4 OUT3 OUT2 OUT1

控制电机输出端

OUT1 OUT2 连接左电机

OUT3 OUT4 连接右电机

VDD驱动电源输入端输入电压6~12V

连接12V蓄电池的正极

GND公共接地端

连接12V蓄电池的负极

+5V稳压可以控制系统供电

{58%： 连接树莓派的+5V电源输出引脚}

IN4 IN3 IN2 IN1 逻辑输入端

分别与树莓派的37，38，13，15号引脚相连 控制这些引脚输出高低电平组合

IN1 =0 IN2=1|| IN3 =0 IN4=1

对应电机即可正转

IN1 =1 IN2=0|| IN3 =1 IN4=0

对应电机即可反转

IN1 =0 IN2=0|| IN3 =0 IN4=0

对应电机即可停止

IN1 =1 IN2=1|| IN3 =1 IN4=1

对应电机即可停止

综上所述，驱动电机的一个输入端必须要高，而其他输入端的相对而言要低。 高电平输入端决定电流方向。

本文使用L298N驱动板，但对于如何选择驱动GPIO，可采取两种方案：

一、使能端高电平，但为驱动器输入选择安全的电机GPIO引脚。

二、从安全的电机GPIO驱动使能端，启动后配置其他GPIO引脚。

第一种选择不能使用使能端。 因此，在启动时， 对于电机控制而言所有的GPIO输入引脚都是安全的。 缺点是四个输入控件都需要从安全GPIO池中移除。 如果需要驱动多台电机，选择将变得有限。

第二种方法是在L298N驱动器的两个使能端上使用GPIO安全电机引脚。 这种方式下，在启动过程中，使能端被拉低，不管输入引脚是何状态，禁用电机控制。 使得可以灵活地选择其他GPIO引脚作为输入信号。 本文选择第二种方法。

由此，解决树莓派小车方向的控制问题。

3.3 树莓派GPIO引脚安全输入电压

{83%： 从树莓派向超声波传感器的触发引脚给出一个输入信号时，发射器就会发射声波脉冲，} {63%： 发出的声波脉冲被物体表面放射回来后，可从回波引脚接收到回波脉冲。} {89%： 然后可以计算回波的返回时间

，之后就可以计算距离。} 距离计算公式：

（3.2）

树莓派上5V的供电引脚。 接到超声波的VCC引脚上，为其供电。 {56％：而作为回应，超声波传感器会产生5V的回波信号输给树莓派，但是，树莓派需要在GPIO引脚上加上3.3V的电平才能安全工作。} {77％：根据基尔霍夫电压和电流定律，将5 V划分为3.3 V和1.7 V，就可以让回波从超声波传感器输出，} {86％：并让其输入到与超声波传感器连接的树莓派的 GPIO引脚上。} 如图3.5所示。

图3.5 分压器

输出电压计算公式：

（3.3）

已有1k电阻，即R1＝1k，而Vin＝5V Vout＝3.3V，根据公式（3.3）计算出分压电阻：

{91％：经过上述计算，需要两个电阻以制作将电压值从5V变为3.3V的分压器，这两个电阻的值分别是1k和2k。}

3.4处理器负载均衡

为了解决负载均衡问题，充分利用CPU资源，提高CPU的使用率，同时由于处理大量的IO操作需要花费大量的时间等等，采用多线程的方式去同时完成几件事情而不互相干扰。 比如: {53％：读写文件，视频图像的采集，处理，显示，保存等。}

Python中使用线程有两种方式： 函数或者用类来包装线程对象。 {43％：在本论文中，采用第二种方式，使用Threading模块创建线程，直接从Thread类继承，然后重写__init__方法和run方法。}

{91％：当并发执行的的线程共享数据时，各线程会改写共享的数据，由于CPU调度顺序的不确定性，造成线程运行的结果不确定性。} {92％：为了保证结果的可再现性，各线程执行序列必须加以限制一保证互斥地使用临界资源，相互合作完成任务，} 在 Python中使用 threading. Event()方法提供的释放和申请信号量的方式处理线程同步互斥机制。

{64％：如图3.6所示，服务器线程流程图：}

图3.6 服务器线程框架图

3.5 网络传输

为保证数据传输能够满足稳定、可靠、实时的要求，本文采用有连接的、可靠的TCP通信方式。 如图3.7所示。

{66％：在树莓派上，使用无线网卡模块。} 当客户机开启时，经过网络配置的树莓派会自动连接客户机的

无线网络。

在Python编程中，导入模块socket，集成了用于网络编程的类和方法。

from socket import *

图3.7 TCP通信

4 实现

4.1 控制伺服电机和直流电机模块

{60%：与树莓派连接的控制电路如图4.1所示：} 两个伺服电机分别安装在云台的X轴平面和Y轴平面上，带动云台上的摄像头拍摄视角在水平面和垂直平面上移动。 {47%：L298N电机驱动模块输出端与直流电机相连，输入端与树莓派相连，实现小车的转向功能。}

图4.1 控制电路图

定义和封装了一个类(模块)，伺服电机模块 class SG90，模块中首先完成模块的初始化， {53%：然后定义了向上转动方法 up，向下转动方法 down以及向向左 left和向右转动方法 right，} 最后定义一个命令调用方法 commend，为外部程序调用和传参控制提供接口。

根据文章3.1，使用Python GPIO模块中的ChangeDutyCycle()方法，传入给定的pwm脉冲占空比值，伺服电机就会转到与其对应的角度。

1.导入GPIO模块。 定义控制信号引脚编号如图4.1 树莓派的26号、29号引脚分别连接水平面与垂直平面的伺服电机，进而控制云台摄像头。

import RPi.GPIO as GPIO

import time

POUTX = 26

POUTY = 29

DC_SET = 0.8

2.创建对象。 {41%：利用GPIO.PWM类定义一个对象，控制产生脉冲宽度调制信号。} 以控制水平面的伺服电机为例，设置指定引脚POUTX，定义pwm信号频率为50Hz，返回一个pwm信号实例pwm_x。 start启动信号。

self.pwm_x = GPIO.PWM(POUTX， 50)

self.pwm_x.start(0)

3.自定义函数方法。 根据传入的参数，伺服电机转到指定的角度，如产生一个控制向左转的方法 left(key)，参数 key为键盘的字符按键， {44 %： 若每按键为 L，产生 pwm信号的占空比值就会相应增加，驱使伺服电机向左转。 }

def left(key):

if(key == ’ L ’):

dc_x += DC_SET

pwm_x.ChangeDutyCycle(dc_x)

time.sleep(0.2)

pwm_x.ChangeDutyCycle(0)

为控制方向，底盘的两个主动轮被连接到2个独立的电机。 {43 %： 电机驱动芯片L298N可以驱动这两个电机同步旋转，当两个电机同时旋转时，则可使小车前进或后退。 } 软件编程与伺服电机类似，控制直流电机，本文中也定义了一个类class CarCtrl，类中定义了控制电机转动方向的方法，即控制小车行驶的方法。

根据文章3.2，控制电机转动方向，需要在树莓派与电机之间连个一个 L298 N电机驱动板， 在图4.1中，树莓派的10号、11号引脚连接驱动板的 INT1， INT2引脚， {55 %： 通过输入到该两个引脚的高低电平组合，控制M1电机转向。 }

定义两个方法right、back，功能别是电机M1正转和反转，而方法cmmend(key)，根据字符键盘的输入，传入到参数key，为外部调用类成员函数提供接口。

import RPi.GPIO as GPIO

INT1 = 10

INT2 = 11

def right(): # M1 正转

GPIO.output(INT1，GPIO.LOW)

GPIO.output(INT2，GPIO.HIGH)

def back(): #M1 反转

GPIO.output(INT1，GPIO.HIGH)

GPIO.output(INT2, GPIO.LOW)

def commend(self, cmd):

if(cmd == 'S'):

back()

elif(cmd == 'D'):

right()

4.3 控制传感器

4.3.1 超声波传感器

{47%：超生波传感器HC_SRO4与树莓派电路图如图4.2所示：}

图4.2 树莓派与HC_SR04

已知可超声波传感器的工作原理，计算出了距离公式，明确了连线图，开始编写Python脚本程序:

1.初始化配置：

{60%：在Python环境中引入或调用GPIO库和time库，time库为Python代码提供即时和延时功能，采用BOARD模式调用树莓派的GPIO引脚。}

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

POUT = 21

PIN = 22

2.将GPIO设置为默认模式：

{86%：将传感器的触发引脚、回波引脚分别与树莓派的21号和22号引脚连接起来。} {94%：所以，触发引脚是树莓派的输出引脚，也是传感器的输入引脚，回波引脚是树莓派的输入引脚，在发送触发脉冲后，回波引脚会从传感器得到响应。}

GPIO.setup(POUT，GPIO.OUT)

GPIO.setup(PIN，GPIO.IN)

GPIO.output(POUT，False)

3.发送和接收脉冲：

对测量距离的超声波传感器的时序图分析，在前一步中将触发引脚 POUT默认设置为0，{98％：然后从树莓派向传感器的触发引脚发出一个10 us脉冲，一旦传感器接收到这个触发信号，}{93％：它就会从内置的发送器中发出40 kHz的脉冲（超声波脉冲：} 包含8个脉冲）。 如图4.3所示。

{76％：图4.3 超声波传感器时序图}

{76％：在超声波脉冲信号发出之后，等待回波信号。}{87％：每个回波信号的长度为进行距离的计算提供了计时。}{90％：通过将触发引脚设为持续时间10us的高电平来创建触发脉冲。}

GPIO.output(POUT，GPIO.HIGH)

time.sleep(0.000001) #0.000001s = 10us

GPIO.output(POUT，GPIO.LOW)

{73％：利用以上代码，在内部生成超生波脉冲并将其发送到空气中。}{86％：反射回来的超声波脉冲就形成了回波脉冲，需要对回波脉冲进行检测直到接收到它为止。}{80％：如图4.4所示，回波引脚保持了一段时间的高电平，所以需要计算接收到的回波脉冲驻留为高电平的时间。}{78％：用while循环持续监控回波引脚，time函数得到回波保持高电平的持续时间。}

while GPIO.input(PIN) == 0: #GPIO.input() 读取引脚状态

start_time = time.time()

while GPIO.input(PIN) == 1:

end_time = time.time()

{62％：变量start_time为时间值（时间戳），而当回波引脚的状态从0变为1时，会跳出第一个循环。}{74％：由此得到回波脉冲由低变高的时间。} 接下来，由相同的逻辑获得回波脉冲由高变低的时间戳end_time。 {81％：最终，得到了回波脉冲的开始时间和结束时间，将两者相减得到脉冲的持续时间。}

t = end_time - start_time

4.计算距离：

根据公式（3.2）计算距离： distance＝17150＊t（m）

4.3.2 DHT11温湿度传感器

DHT11传感器使用的信号和1-Wire协议相似，但响应时间不同。 此外，没有设备序列号的支持。 {41%：适用该传感器的供电电压为3V~5.5V，从树莓派的3.3V电源供电，可保证传感器信号水平处于对GPIO的安全范围内。} {45%：与树莓派连接的电路图如图4.4所示，电源引脚外接3.3 V电压，} {42%：接地引脚接共地端，数据引脚接树莓派31号通用引脚。}

{41%：当树莓派在GPIO引脚上监听或DHT11不传数据时，线电压将上浮。} 出于此原因，需要在电源线与引脚间串接一个5千欧姆的电阻，使线电压提高到3.3V的稳定水平上。 当树莓派引脚与传感器处于活跃状态时，提供给GPIO引脚或传感器的电流小于1mA。

图4.4 树莓派与DHT11

1、起始信号： 无信号请求时，线路由于上拉电阻而高电平闲置； 有起始信号请求时，树莓派需要将线路拉至低电平18 ms，以发出读取请求，然后释放总线， 允许线路返回高电平状态，如图4.5，代码段中， setup与output设置31号引脚为低电平输出， {43%： sleep延时0.02 s，最后输出高电平，将线路拉回至高电平状态。}

```
GPIO.setmode(GPIO.BOARD)

GPIO.setup(31， GPIO.OUT)

GPIO.output(31， GPIO.LOW)

time.sleep(0.02)

GPIO.output(31， GPIO.HIGH)
```

图 4.5 主机发送起始信号

2. THD响应信号、准备信号： 在停顿20us~40us之后，传感器做出响应，控制线路状态保持80us的低电平，然后允许线路返回一个80us的高电平状态。 {57%：这会发出传感器的意图以反馈数据，如图4.6所示。} 代码段中，setup设置31号引脚为输入，input读取引脚状态，当为低电平，进入第一个while循环，直到为高电平，进入第二个while循环，直到重新返回高电平状态。

```
GPIO.setup(31， GPIO.IN)

while GPIO.input(channel) == GPIO.LOW:

continue

while GPIO.input(channel) == GPIO.HIGH:
```

continue

图 4.6 响应信号

3. DHT数据信号： 40位的数据信息会被记录至总线上，每个传感器数据位以转变低电平开始，然后转换高电平。 如图4.7，每个数据位以转变为低电平开始，持续50us。 最后一位之后的最后一次转变为低电平也持续50us，在完成一位的低到高信号转变后，如果高电平持续仅仅26us~28us，则该位变为0。 相反，高电平持续70us该位为1。 当作为下一位开始的从高到低的转变发生时，每个数据位结束。 代码段中，进行40次循环，对应40次数据的读取，最终的数据存储在data变量中。

```
while j [ 40:

k = 0

while GPIO.input(channel) == GPIO.LOW:

continue

while GPIO.input(channel) == GPIO.HIGH:

k += 1

if k ] 100:

break

if k [ 8:

data.append(0)

else:

data.append(1)

j += 1
```

图 4.7 数据格式

4.数据切片处理: 代码段中，变量 data中存储了相对湿度、摄氏温度、校验位共40 Bit数据，需要通过切片和移位操作， 将数据分别存储到 humidity(湿度)、 temperature（温度）变量和 check校验变量中。

```
humidity_bit = data[0:  8]

humidity_point_bit = data[8:  16]
```

temperature_bit = data[16: 24]

temperature_point_bit = data[24: 32]

check_bit = data[32: 40]

for i in range(8):

humidity += humidity_bit[i] * 2 ** (7 - i)

humidity_point += humidity_point_bit[i] * 2 ** (7 - i)

temperature += temperature_bit[i] * 2 ** (7 - i)

temperature_point += temperature_point_bit[i] * 2 ** (7 - i)

check += check_bit[i] * 2 ** (7 - i)

5. DHT结束信号： 当传感器带动线路超过50us后，信号传播过程停止。 传感器释放总线，允许线路恢复成空载高运行状态。 总时序如图4.8所示。

图 4.8 总时序图

4.4.3 烟雾火焰传感器

如图4.10，树莓派的32号，33号引脚分别与火焰传感器和烟雾传感器的TTL电平输出口相连。 定义了两个方法isFlame、isFog，input通过读取引脚的输入电平，判断传感器是否探测到烟雾和火焰。

import RPi.GPIO as GPIO

D2 = 33

D3 = 32

def isFlame():

if GPIO.input(D3) == 0: #32号引脚为低电平探测到火焰

return 1

else:

return 0

def isFog():

if GPIO.input(D2) == 0: #33号引脚为低电平探测到烟雾

print ”D2 == 0”

return 1

else:

return 0

{53%： 图4.9 树莓派与火焰、烟雾传感器}

4.4.4 气压传感器

使用I2C总线协议来控制BMP180压强传感器，并且传感器内的寄存器和RAM可被随机写入。 {47%： 先要指定一个起始寄存器地址，然后一个或多个字节被写入。} 寄存器地址随每个字节自动递增，并回到0。 BMP180从设备会确认（ACK）每个被收到的字节，一直持续到主设备（树莓派）写一个停止位（P）。 发送的每一个字节总是外设的I2C地址，该地址与所选择的外部设备的寄存器地址不是同一概念。

表 4.10所示，E2PROM中的寄存器，通过这些寄存器的地址，来访问它们的值，这些值作为校准使用。

表 4.10 E2PROM中的寄存器表

寄存器地址

校准值大小

校准值

AC1(0xAA，0xAB)

(16 bit)

408

AC2(0xAC，0xAD)

(16 bit)

-72

AC3(0xAE, 0xAF)

(16 bit)

-14383

AC4(0xB0, 0xB1)

(16 bit)

32741

AC5(0xB2, 0xB3)

(16 bit)

32757

AC6(0xB4, 0xB5)

(16 bit)

23153

B1(0xB6, 0xB7)

(16 bit)

6190

B2(0xB8, 0xB9)

(16 bit)

MB(0xBA, 0xBB)

(16 bit)

-32768

MC(0xBC, 0xBD)

(16 bit)

-8711

MD(0xBE，0xBF)

(16 bit)

2868

根据 I2 C总线协议，首先需要明确 I2 C设备的总线地址，该设备的 I2 C总线地址为 BMP180_ I2 CADDR=0 x77，然后设置设备的工作模式： 超低功耗、标准、高分辨率、超高分辨率，如表4.11所示，其次读取原始数据， {41 %： 最后根据设备中 E2 PROM寄存器的数据，补偿校准原始传感器数据。 }

表4.11 工作模式

总线地址

BMP180_I2CADDR 0x77

工作模式

BMP180_ULTRALOWPOWER 0

BMP180_STANDARD 1

BMP180_HIGHRES 2

BMP180_ULTRAHIGHRES 3

超低功耗

标准

高分辨率、

超高分辨率

控制寄存器地址

BMP180_CONTROL 0xF4

数据寄存器地址

BMP180_TEMPDATA 0xF6

对BMP180的编程设计如图4.12所示：

图4.12 bmp180编程流程

1.导入相关库/模块。 {42%：在程序中使用Python中的系统总线库smbus，访问I2C设备总线。}

import smbus#导入系统总线模块

2.定义一个方法pressure_data。 使用 smbus模块中的 write_ byte_ data方法和 read_ byte_ data方法分别写、读设备中的寄存器值， 完成设备的的初始化操作，最终读取到原始（未补偿）的压强值并存储到 raw变量中。

def pressure_data():

self._write_byte(BMP180_CONTROL， BMP180_READPRESSURECMD + (self._mode [[6))

MSB = self._read_byte(BMP180_PRESSUREDATA)

LSB = self._read_byte(BMP180_PRESSUREDATA+1)

XLSB = self._read_byte(BMP180_PRESSUREDATA+2)

raw = ((MSB [[16) + (LSB [[8) + XLSB)]] (8 - self._mode)

3.计算真实压强值。 由函数pressure_data得到未补偿的气压值raw，根据E2PROM寄存器中的补偿值，校准更新气压数据pressure。

6.计算海拔高度。 根据海拔计算公式（2.2），由大气压计算绝对海拔高度altitude。

altitude = 44330.0 * (1.0 - pow(pressure / 101325.0， (1.0/5.255)))

4.4 视频传输

4.4.1 USB摄像头

{46%：USB网络摄像头连接树莓派的USB接口，树莓派会自动检查摄像头设备驱动，并在/dev目录下生成一个video0的文件。} 该文件与摄像头对应，操作这个文件相当于操作这个摄像头，如打开或关闭文件，相当于打开或关闭摄像头。

{43%：在Python编程中，使用cv2库下面的cv模块操作了video0文件，所以可以使用该模块对摄像头进行操作；} socket模块用于网络编程； Image模块对图像的基本处理，StringIO库对字符串型数据进行缓存。

1.导入相关库/模块。

from socket import *

import cv2.cv as cv， Image， StringIO

2.创建摄像头对象： CaptureFromCAM创建一个源于USB摄像设备的捕捉器(对象)，SetCaptureProperty设置捕捉器图像的分辨率640x480。

capture = cv.CaptureFromCAM(0)

cv.SetCaptureProperty(capture， cv.CV_CAP_PROP_FRAME_WIDTH， 640)

cv.SetCaptureProperty(capture， cv.CV_CAP_PROP_FRAME_HEIGHT， 480)

3.建立TCP网络通信： socket创建监听套接字 sockfd--] bind绑定主机 IP地址与端口 PORT--] listen启动监听--] accpet等待连接， {46％：客户机发起连接请求，返回一个通信套接字 connfd和一个客户机地址 addr。}

sockfd = socket(AF_INET， SOCK_STREAM)

sockfd.bind(IP， PORT)

sockfd.listen(5)

connfd， addr = sockfd.accept()

4.压缩图像： QueryFrame从摄像头捕捉器抓取并返回一帧pic图像。 {50％：因为原始的数据图像分辨率为640 x480，计算出一帧图像约900 kB，图像数据太大，} 导致帧传输速率差，视频播放时有严重的延迟，所以使用fromstring将图像压缩为 JPEG格式， {42％：减少了图像的大小，提高了帧速率，保证了远端视频播放的流畅性。}

pic = cv.QueryFrame(capture)

pil = Image.fromstring(＂RGB＂， cv.GetSize(pic)， pic.tostring())

5.保存图像： {43％：StringIO得到一个内存文件，save将JPEG格式的图像保存在内存文件中。} {41％：getvalue得到内存文件(图像)中的数据，将图像保存在内存文件，避免了从内存到硬盘的操作，提高了程序运行的效率。}

file = StringIO.StringIO()

pil.save(file， format = ＂JPEG＂)

jpeg = file.getvalue()

6.传输图像： 因为远端接收图像数据流时遇到字符＇\n＇，会刷新一次内存，导致接收不完整图像。 为了实现一帧图像一次性传输与接收，用 replace将图像流中的＇\n＇替换为其它字符如＇\-n＇， 最后在整个图像流

末尾添加字符 ’ \ n ’ 并由 sendall发送，保证远端没接收一帧图像， 仅刷新一次内存。

transfer = jpeg.replace(” \n ” ， ” \-n ”)

connfd.sendall(transfer + ” \n ”)

4.4.2 CSI摄像头

对于CSI接口的摄像头模块，可利用Python的picamera模块对其进行操作，如完成一次图像的发送。 { 52 %： 代码段中不再采用 USB摄像头的视频图像传输方式，而是先捕捉到一帧 JPG格式的图像， } { 49 %： 然后获取图像大小，其次发送图像大小，最后循环发送图像数据，保证远端接收图像的完整性。 } 由于没有进行图像压缩，与USB接口的摄像头相比，图像帧速率较低，但图像清晰度较高。

1.导入模块picamera。 { 41 %： PiCamera获取摄像头一个实例camera，resolution设置摄像头图像的分辨率 300x200。 }

import picamera

camera = picamera.PiCamera()

camera.resolution = (300， 200)

2.捕捉和发送图像。 capture捕捉一张图像，并命名为imgCSI.jpg。 getsize获取图像大小，pack将图像数据大小打包，并将其指定为长整型的数据类型，send发送图像大小fhead。 open打开图像，for循环内读取并发送图像数据。

camera.capture(” imgCSI.jpg ”)

fhead=os.path.getsize(’ imgCSI.jpg ’)

fhead = struct.pack(’ l ’ ， fhead)

connfd.send(fhead)

with open(’ imgCSI.jpg ’ ， ’ rb ’) as fp:

for data in fp:

connfd.send(data)

4.5 客户端GUI编程设计

Python是脚本语言，Qt是开发GUI应用程序的最佳库。 Python和Qt的结合，PyQt，可以在任何支持的平台上开发应用程序和运行。 本文在虚拟机Ubuntu系统下，使用PyQt开发工具，进行客户端的GUI编程设计，设计流程

如图4.13

图4.13 PyQt4 GUI设计框图

4.5.1 MainWindow类

在主线程中，自定义一个类 MainWindow，该继承于 QWidget， lcd与 label为声明的类的全局属性变量列表（公有成员变量）， {48 %： __init__为初始化函数（构造函数），构造函数中调用了父类 QWidget的构造函数。}

```
class MainWindow(QWidget):

lcd = [ ' lcd1 ' ,  ' lcd2 ' ,  ' lcd3 ' ]

label = [ ' label_camera ' ,  ' label_distance ' ,  ' label_warning ' ,  ' label_humi ' ]

def __init__(self,  parent = None):

QWidget.__init__(self,  parent):
```

类中的其它主要的成员函数及函数功能如表4.14所示：

表4.14 类成员函数及功能

createLayout(self)

创建界面布局并初始化

connectToHost(self)

连接主机，并创建子线程

webShow(self)

组合一个浏览器类

sendYeeLink(self)

将数据上传至YeeLink平台

keyPressEvent(self)

重写按键函数，启动键盘监听

playButtonCSIClicked(self)

CSI摄像头开启按钮

playButtonUSBClicked(self)

USB摄像头开启按钮

flushPicCSI(self，val)

CSI摄像头显示

flushPicUSB(self)

USB摄像头显示

showTime(self)

获取实时时间

playWarningSound(self)

警告危险

createSinglSlot(self)

创建信号与槽

createLayout，布局三个数码管，用于显示了温度、大气压强、海拔高度、空气湿度参数，并且设置了相应的显示样式如图。 以显示温度传感器的数据为例，代码片中，QLCDNumber返回一个数码管控件，QLabel用于显示温度和文本信息。

图4.15 数码管显示

MainWindow.lcd[0] = QLCDNumber(self)

self.label1 = QLabel(u”温度: ”)

self.label1_1 = QLabel(u”C”)

布局三个警告信息图标，实际上是在按钮上设置了图标： 如图4.16，当传感器探测到火焰和烟雾时，相应的图标会发生改变： 如图4.17。 以第一个图标为例，QIcon加载一张图标，图标路径为./image/ico/safe.jpg，QPushButton返回一个按钮实例，setIcon将图标添加在按钮上。

图4.16 安全图标

图4.17 警告图标

self.icoSafe = QIcon(” ./image/ico/safe.jpg ”)

self.icoNoSafe = QIcon(” ./image/ico/nosafe.jpg ”)

self.buttonSafe = QPushButton()

self.buttonSafe.setIcon(self.icoSafe)

布局两个标签，用于显示视频图像。 以 USB摄像头的视频图像为例， QLabel返回一个标签实例，为初始化界面，先用 QImage返回一个图像对象 img， load加载一张图像到 img， scaled改变图像分辨率， setPixmap显示在标签上。

MainWindow.label[0] = QLabel()

self.img = QImage()

self.img.load(” ./image/swpu.jpg ”)

self.img= self.img.scaled(700， 500， Qt.KeepAspectRatio);

MainWindow.label[0].setPixmap(QPixmap.fromImage(self.img))

总体布局如图4.18所示：

图4.18 PyQt GUI设计的界面

connectToHost，根据输入框获取到服务器 IP和端口，点击登录，发送连接服务器请求，并创建三个子线程实例 recvthread、 picUSBthread、 picCSIthread， 与其对应的三个线程类 recvThread、 picUSBThread、 picCSIThread在 MainWindow类外创建。

HOST = self.lineEditIP.text()

PORT = int(self.lineEditPort.text())

self.sockfd = socket(AF_INET， SOCK_STREAM)

self.sockfd.connect((HOST， PORT))

self.recvthread = recvThread() #字符串线程(接收传感器信息)

self.picUSBthread = picUSBThread() #USB图像线程(接收图像信息)

self.picCSIthread = picCSIThread() #CSI图像线程

webShow，组合一个浏览器类，点击打开浏览器，弹出一个YeeLink网页，便于查看上传至YeeLink平台的数据。如图4.19所示：

self.view = MyBrowser()

self.view.show()

图4.19 浏览器

sendYeeLink，用于将数据上传至YeeLink平台。 在该平台上，一个用户仅拥有一个密钥U-ApiKey，一个设备（树莓派）只能有一个设备ID，但可有多个传感器ID。 以上传温度传感器为例，树莓派的ID为356534，温度传感器的ID为407095，根据用户密钥和ID号，使用post发送http请求，上传经过打包的数据。

apiheaders = { ' U-ApiKey ' : ' 8b0df11c296b573b852b31a417a24e30 ' , ' content-type ' : ' application/json ' }

apiurlTemp = ' http: //api.yeelink.net/v1.0/device/356534/sensor/407095/datapoints '

payloadTemp = { ' value ' : temp_2}

requests.post(apiurlTemp, headers=apiheaders, data=json.dumps(payloadTemp))

keyPressEvent，重写了父类QWidget中的keyPressEvent方法，启动键盘监听，处理响应事件。 例如当按下w键时，向服务器发送一个字符命令w，控制树莓派小车向前运动。

if event.key() == Qt.Key_W:

self.sockfd.send(' w ')

{ 41 %： playButtonCSIClicked，槽函数，当点击视频播放按钮时，会发送一个clicked信号，触发该槽函数。 } 槽函数中，改变了按钮得图标，启动了线程picCSIthread。 该槽函数与playButtonUSBClicked类似。

self.isCSI = True

self.buttonCSI.setIcon(self.ico5)

self.picCSIthread.start()

flushPicCSI，槽函数。 当线程picCSIthread接收到一帧图像时，向MainWindow类发送一个自定义信号savePicCSI，进而触发该槽函数，刷新标签上的图像。 该槽函数与flushPicUSB类似。

self.imgCSI.load(filename)

self.imgCSI = self.imgCSI.scaled(300， 200， Qt.KeepAspectRatio)

self.labelCSI.setPixmap(QPixmap.fromImage(self.imgCSI))

showTime，首先time.time()获取获取当前时间戳，然后localtime()将以秒为单位的时间戳转化为一个时间元组，最后使用strftime()将时间元组转化为特定的字符串。

now = int(time.time())

timeArray = time.localtime(now)

currentTime = time.strftime(”%Y-%m-%d %H: %M: %S”，timeArray)

playWarningSound，槽函数。 首先当火焰、烟雾传感器探测到火焰或烟雾时，线程recvThread会收到一个标志，然后线程向该槽函数发送信号warningSound，触发槽函数，发出警告声且使图标发生改变。

os.system(”mpg321 ./sound/beez.mp3 ”)

self.buttonSafe.setIcon(self.icoNoSafe)

createSinglSlot，连接信号与槽，在MainWindow类中： 信号和槽如表4.20：

表4.20 信号槽函数

产生信号的对象

信号

登录按钮

loginButton.clicked

connectToHost

重设按钮

setButton.clicked

reSet

浏览器按钮

connWeb.clicked

webShow

定时器

yeeLinkTime.timeout

sendYeeLink

USB视频图像播放按钮

playButton.clicked

playButtonUSBClicked

拍照按钮

saveVideoButton.clicked

saveVideoButtonClicked

CSI视频图像播放按钮

buttonCSI.clicked

playButtonCSIClicked

警示按扭

recvthread.warningSound

playWarningSound

线程1

picthread.savePic

flushPicUSB

线程2

picCSIthread.savePicCSI

flushPicCSI

4.5.2子线程类

主线程即MainWindow类创建了3个子进程，每个子进程都继承于QThread，重写了run成员函数，实现与服务器对应的子线程建立连接，得到通信套接字sockfd。

class thread(QThread):

def __init__(self):

super(thread，self).__init__()

def run(self):

self.sockfd = socket(AF_INET，SOCK_STREAM)

self.sockfd.connect((HOST，PORT))

1. class picCSIThread，在子线程1中，recv接收传感器数据data，unpack将数据解包，’i’表示解包的数据类型为整型，display将数据显示在数码管上。

data = self.sockfd.recv(DATA_SIZE)

temp_2，humi，pressure，altitude，distance，flame，fog = struct.unpack(’iiiiii’，data)

MainWindow.lcd[1].display(pressure)#压强 MainWindow.lcd[2].display(altitude)#海拔

MainWindow.lcd[0].display(temp_2)#温度

MainWindow.lcd[3].display(humi)#湿度

2. class picUSBThread，子线程2中，接收USB摄像头图像信息。 代码段中，makefile将套接字sockfd与文件关联起来，以操作文件的方式操作套接字，如使用readline读取文件中的数据。 replace还原’\n’。 {50%：StringIO将数据以数据流的方式存入到内存。} Image返回一个PIL对象pil。 CreateImageHeader，创建IplImage图像头，设置图像分辨率640x480、深度IPL_DEPTH_8U以及通道数3。 SetData填充数据。 SaveImage保存图像，最后向MainWindow类发送图片保存信号savePic。

f = self.sockfd.makefile()

msg = f.readline()

jpeg = msg.replace(”\-n”，”\n”)

buf = StringIO.StringIO(jpeg[0: -1])

buf.seek(0)

pil = Image.open(buf)

img = cv.CreateImageHeader((640，480)，cv.IPL_DEPTH_8U，3)

cv.SetData(img，pil.tobytes())

buf.close()

cv.SaveImage(”./image/imgUSB.jpeg”，img)

self.savePic.emit()

3. class picCSIThread，子线程3中，接收CSI摄像头数据，对图片中的人脸进行识别。 {48%：首先接收的数据是图像的大小，然后进入while循环接收图像数据，直到接收到一帧的图像，} 将接收到图像通过函数face_detect进行人脸识别，最后向MainWindow类发送一个带参信号savePicCSI。

filesize = sockfd.recv(FILEINFO_SIZE)

filesize = struct.unpack(’l’，filesize)

restsize = filesize[0]

with open(’./image/imgCSI.jpg’，’wb’) as fp:

while True:

if restsize] BUFFSIZE:

filedata = sockfd.recv(BUFFSIZE)

else:

filedata = sockfd.recv(restsize)

if not filedata: break

fp.write(filedata)

restsize = restsize - len(filedata)

if restsize == 0: break

if face.face_detect(): #人脸识别函数face_detect()

self.savePicCSI.emit(1)#发送图片保存信号

else:

self.savePicCSI.emit(0)

4.5.3 人脸识别

总体流程： {40%：加载原图--]转化为灰度图（有利转化原图像的颜色空间并存储图像结果）--]灰度直方图均匀化（灰度直方图均衡化对图像增强的效果高，} {49%：对图像细节部分能起到明显的突出增强效果）--]得到人脸数组并画矩形框。}

1.导入相应模块。 cv2库以及库中的cv模块，用于视觉处理的工具。

import cv2

import cv2.cv as cv

2. haar分类器。 下载分类器文件，文件名后缀以.xml结尾，捕捉人脸特征和手机特征。

face_cascade = cv2.CascadeClassifier(’./data/lbpcascades

/lbpcascade_frontalface.xml ’)

phone_cascade = cv2.CascadeClassifier(’./data/lbpcascades/iphone.xml ’)

3.定义函数face_detect。 在子线程3中将会调用该函数，

def face_detect():

img = cv2.imread(” ./image/imgCSI.jpg ”)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)#转换为灰度图

gray = cv2.equalizeHist(gray)#直方图均衡处理

faces = face_cascade.detectMultiScale(gray)#通过分类器得到faces

if len(faces) == 0:

return 0#检测失败返回0

conut = 1

cv2.putText(img, ” FPS: 4 ” , (10, 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

cv2.putText(img, ” NUM: ” +str(len(faces)), (170, 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

for (x, y, w, h) in faces: #对每一个人脸画矩形框

cv2.rectangle(img, (x, y), (w+x, h+y), (0, 255, 0), 2)

cv2.putText(img, ” Face. ” +str(conut), (x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

conut += 1

cv2.imwrite(” ./image/imgCSI_detect.jpg ” , img)

return 1 #检测成功返回15 测试

{ 56 %： 1、在终端启动小车服务器，等待客户端连接。 } 如图5.1所示：

图5.1 服务器运行

{ 42 %： 2、在虚拟机的 Ubuntu系统下运行客户端应用程序，点击登录按钮，连接树莓派服务器， } 并启动键盘监视事件，输入按键可对小车行驶方向和舵机（云台）的控制: W--前进，S--后退，A--左转，D--右转； I--向上，K--向下，J--向左，L--向右。

图5.2 连接服务器

3、连接服务器成功，开始接收小车上与树莓派连接的传感器数据，并用数码管显示。 如图5.3所示：

图5.3 显示的各个传感器数据

4、点击/按钮，开始接收USB摄像头的视频数据。 { 44 %： 如图5.4所示，视频上显示距离前方障碍物的距离，以及时间。 } { 58 %： 距离由超生波传感器探测得到。 }

图5.4 接收USB接口的摄像头图像

5、点击/按钮，开始接收CSI摄像头的视频数据。 { 42 %： 如图5.5所示，摄像头的视频数据经过人脸识别程序处理后，在人脸周围画出矩形框。 }

图5.5 CSI接口的摄像头用于人脸识别

6、将所有数据上传至互联网平台。 如图5.6~5.8所示：

//

图5.6 摄像头图像

//

图5.7 温度、压强

//

图5.8 海拔、湿度

7结论与建议

为了完成本次毕业设计，自学了一门的编程语言Python，并结合实际，把程序设计与硬件电路紧密地结合在一起，编写程序使一些硬件设备能正常工作。 比如在文章的4.1，用Python的扩展库python-gpio库，轻松访问树莓派的GPIO引脚； 在文章的4.5.1，用强大的PyQt4扩展包，设计了客户端界面程序； 在文章4.5.6，利用cv2库，对图像进行简单的处理。 在了解了一些硬件电路的背后的原理之后，再结合以前对模拟电路和数字电路的学习，这对编写程序实现各个功能提供了可行性。 如在文章4.3.2和4.4.4，使用了几种常用的通信协议 one- wire、 I2 C、 SPI， DHT11温湿度传感器采用第一种通信方式，这种方式是无同步时钟的，需要通过特定的约定来进行通信， {41%：有时会出现数据紊乱，因此需要对 DHT11采集的数据进行错误判断；} LCD显示器通过SPI(串行外设接口)与数据总线连接，在开启服务器程序时，显示西南石油大学校徽； BMP180气压传感器则通过I2C与数据总线相连，因为该传感器需要对采集的数据进行编程处理，所以是整个程序最为复杂的一部分之一。

在 Python网络编程这方面，遇到许多困难，比如采用什么方式传送数据，一开始，使用 json模块打包发送的数据， 但发现，只能打包字符串类型格式的数据，对于其它类型的数据，则需要类型转换，这大大减低了程序执行的效率， 最后，采用 struct模块，例如，在文章4.5.3，使用该模块中的 pack函数对数据进行打包， 接收方通过套接字接收到数据后，再使用 unpack函数对数据进行解包，期间在整个过程中，不仅不需要对数据类型进行任何的转化， 而且执行效率也大大加快。

对视频监控，使用过两种方式motion，mjpg-streamer， motion图像的实时性差，传输太卡顿； {96%：而streamer传输视频流很快，几乎不卡顿。} 这两种方式实质是两种开源软件，只需编译、安装、执行即可， 操作方便，但无法嵌入到自己的程序中，只能另外开辟进程独立运行，然而这样做产生一个新的问题， 有些自带功能不适用，且太占处理器资源。 为了提高执行效率，兼容自己的程序，使用Pyhong提供cv2库、Image库、socket库处理视频图像和传输图像。 在文章4.4.1，对于 USB接口的摄像头，先将捕获到的图像压缩成 JPEG格式， 再保存内存文件中，然传输给接收方，传输较为流畅，图片清晰，度较差； CSI接口的摄像头，则直接使用picamera模块捕获图像后再传送，传输的帧数不高，但清晰度较高，高清晰度增强了人脸识别率。

最后，在编写代码方面，为了获得更好的性能，可能还要不断改写代码，删除不必要的变量、循环， 同时还需注意处理中断、定义处理器的休眠时间和看门狗、管理合适的内存片段，以避免程序崩溃。 Python有丰富的代码库，这些代码库都是一些高级开发者编写完成的，所以只要调用一个函数，就能在后台执行所有的内置调用，

有了这些代码库，就能编写出更健壮的代码，减少常犯的错误。

谢辞

本次毕业设计论文是在周云旭老师悉心指导下，在师兄、师姐们的帮助下所完成的。 {47%：首先，非常感谢周老师对我本次毕业设计所给予的帮助。} 在论文的准备阶段，周老师向我提供了大量的参考文献，师兄、师姐们也向我阐述了一些传感器的工作原理。 在辅助师兄们编写程序阶段，查阅了很多的相关知识，在程序初稿完成后的调试阶段，他们也给了我很多的帮助和指点，使自己对程序的所采用的算法有了详细地认识。 在论文最后部分，师姐向我提供了很多的基础数据，并教我如何分析、处理实验点，使得论文中的实验部分更具有准确性。

在完成本次毕业设计的过程中，我学会了许多编程和专业知识，这弥补了过去自己在相关知识学习上所存在的缺陷，也让我懂得了研究问题要从基础开始，分析处理问题一定要抓核心。 因此，在本次毕业论文将要止笔之际，谨向周老师和实验室全体师兄、师姐们致以最衷心的感谢！

{70%：最后，祝愿周老师身体健康，工作顺利，阖家欢乐！}

参考文献

[1]Mark Summerfield. Rapid GUI Programming with Python and Qt[M].Publishing house of electronics industry, 2016.8.1.

[2]赵之珩. 勇气号和机遇号[J]. 青少年科技博览: 中学版, 2004(9): 4-5.

[3] 李向东, 鲁守银, 王宏, 等. 一种智能巡检机器人的体系结构分析与设计[J]. 机器人, 2005, 27(6): 502-506.

[4]矫德余. 基于嵌入式系统的智能巡检机器人研制[D]. 中国石油大学, 2010.

[5] 肖鹏, 王海鹏, 曹雷, 等. 变电站智能巡检机器人云台控制系统设计[J]. 制造业自动化, 2012, 34(1): 105-108.

[6] 王建元, 王娴, 陈永辉, 等. 基于图论的电力巡检机器人智能寻迹方案[J]. 电力系统自动化, 2007, 31(9): 78-81.

[7]蔡焕青, 邵瑰玮, 文志科, 等. 变电站智能巡检机器人应用研究现状[C]. 2015.

[8]Keerthana R , , Jayashree. N R J N R. Ontology based Automatic Module Generation from E-book[J]. International Journal of Computer Applications, 2015, 120: 13-16..

[9]黄雪梅, 范强, 魏修亭. 舵机控制用PWM信号的研究与实现[J]. 微计算机信息, 2010, 26(5): 28-30.

[10] 李方旭, 马彬瀚, 丁伟, 等. 基于HC-SR04超声波传感器的智能避障小车设计[J]. 科技创新与应用, 2016(34): 26-27.

[11] 林倩. DHT11数字温湿度传感器通信协议的IO模拟[J]. 信息通信, 2017(1): 206-207.

[12]Gary Bradski and Adrian Kaebler. Learning OpenCV[M].O ' Reilly Media, Inc. 2008.

[13] Lienhart R, Maydt J. An extended set of Haar-like features for rapid object detection[C]. 2002: I-900-I-903 vol.1.

[14] 林景亮, 唐杰. 一种融合肤色和Haar特征的人脸检测方法[J]. 微型机与应用, 2013, 32(8): 35-37.

[15]贾海鹏, 张云泉, 徐建良. Research on Image Integral Algorithm Optimization Based on OpenCL基于OpenCL的图像积分图算法优化研究[J]. 计算机科学, 2013, 40(2): 1-7.

[16]李文娜. 基于Haar特征级联强分类器和肤色模型的人脸检测[J]. 辽宁石油化工大学学报, 2010, 30(3): 61-64.

[17]孙绪才, L298N在直流电机PWM调速系统中的应用[J].潍坊学院学报, 2009, 9（4）： 1671-4288.

[18]张子木. 基于Arduino的物联网接入技术的研究[D]. 北京工业大学, 2015.

[19]Warren Gay. Mastering the Raspberry Pi[M]. Apress Media, 2014.

[20]Rushi Gajjar. Raspberry Pi Sensors[M]. China Machine Press, 2016.

[21]Bosch Sensortec. BMP180 Digital pressure sensor[R]. Bosch, 2013.

[22] R Swathi, Vision Based Plant Leaf Disease Detection on The Color Segmentation through Fire Bird V Robot[J].2016.